New technologies for web development

Article · January 2010

CITATIONS
12

READS
46,996

4 authors:

Grega Jakus
University of Ljubljana
44 PUBLICATIONS **960** CITATIONS
SEE PROFILE

Matija Jekovec
DHH SpA
1 PUBLICATION **12** CITATIONS
SEE PROFILE

Saso Tomazic
University of Ljubljana
236 PUBLICATIONS **2,184** CITATIONS
SEE PROFILE

J. Sodnik
University of Ljubljana
111 PUBLICATIONS **1,707** CITATIONS
SEE PROFILE

Elektrotehniški vestnik 77(5): 273-280, 2010
Electrotechnical Review: Ljubljana, Slovenija

New technologies for web development

Grega Jakus[1], **Matija Jekovec**[2], **Sašo Tomažič**[1] and **Jaka Sodnik**[1]

[1] *Univerza v Ljubljani, Fakulteta za elektrotehniko, Tržaška 25, Ljubljana*
[2] *Klaro d.o.o. Peruzzijeva 84b, Ljubljana*
E-pošta: grega.jakus@fe.uni-lj.si, matija.jekovec@klaro.si, saso.tomazic@fe.uni-lj.si, jaka.sodnik@fe.uni-lj.si

Abstract. The paper gives an overview of the new features of web technologies. The general idea of the new version of HTML (*Hyper Text Markup Language*), i.e. HTML5, and other tools presented in this paper is the formal specification and the establishment of uniform solutions for technologies and functionalities which have already been in use through various hacks and plug-ins proposed by web developers. Many of these functionalities will now be implemented in browsers. The applications can access these functionalities through newly defined application programming interfaces. The latter include support for multimedia, dynamic graphic rendering, geolocation, multithreading, local data storage etc. HTML5 also introduces semantic markup, which can be used for marking the document structure as well as its elements and data. The new version of HTML enforces strict separation of the page content from its style. The styling can only be done using CSS (*Cascading Style Sheets*) language. The new CSS version, i.e. CSS3, has a modular structure, in which different modules define different styling features. The development cycles of the individual modules are independent as well as their support and implementation in various browsers.

Keywords: World Wide Web, HTML5, JavaScript, CSS3

Nove tehnologije za razvoj na svetovnem spletu

Povzetek. Članek podaja pregled novosti na področju spletnih tehnologij. Eden glavnih ciljev pri razvoju nove različice jezika HTML (*Hyper Text Markup Language*), t. j. HTML5, sta poenotenje in formalna specifikacija nekaterih funkcionalnosti, ki so sicer na spletu že prisotne v obliki lastniških vtičnikov in razvijalskih praks. Številne izmed omenjenih funkcionalnosti bodo odslej vgrajene v spletnih brskalnikih. Različne aplikacije bodo do njih lahko dostopale preko aplikacijskih programskih vmesnikov, definiranih na novo. Ti ponujajo podporo multimedijskim vsebinam, dinamičnemu prikazovanju grafike, geolokacijskim aplikacijam, večnitnosti, lokalnemu shranjevanju podatkov itd. HTML5 uvaja tudi semantično označevanje, ki ga lahko uporabimo za določitev strukture spletnega dokumenta ter pomena elementov in vsebine v njem. Nova različica jezika HTML vsiljuje striktno ločitev zapisa strukture in vsebine spletne strani od njene oblike. Oblikovanje spletne strani je tako odslej mogoče izključno z uporabo jezika CSS (*Cascading Style Sheets*). Nova različica tega jezika, t. j. CSS3, ima modularno zasnovo, pri kateri različni moduli določajo različne oblikovne lastnosti. Razvoj posameznih modulov je med seboj neodvisen, prav tako pa je neodvisna tudi njihova podpora s strani spletnih brskalnikov.

Ključne besede: svetovni splet, HTML5, JavaScript, CSS3

1 Introduction

The concept of the World Wide Web is inseparably tied with the Hyper Text Markup Language (HTML) - the language for describing web pages. HTML uses markup tags for describing structural semantics of a web page by denoting its elements: sections, paragraphs, headings, tables, lists, interactive forms and others. Elements with their corresponding attributes can be nested one in another, forming a typical tree structure. HTML enables also the inclusion of external resources into web documents, such as images, videos and other objects, which also become parts of a web page.

One of the good practices in modern web development are separate definitions of structure and style. The general structure of web pages and their content are defined in HTML, while its final presentation and style are in the domain of CSS (*Cascading Style Sheets*). Such separation enables better flexibility and control over the final appearance of a web page and it also reduces the complexity of HTML record and eliminates the redundancy in style definitions. The separation of the content from the style enables more web pages to share the same style and also a single page to use many different styles at the same time.

Besides CSS, a scripting language JavaScript is often used in combination with HTML. JavaScript is interpreted by a web browser and provides web pages with interactivity and dynamics. The JavaScript code can interact with the DOM (*Document Object Model*) through the various API (*Application Programming Interface*) libraries based on a mechanism of user-triggered events.

Received 19 July 2010
Accepted 23 September 2010

In the 1990's majority of web pages were static and intended primarily for reading and browsing while the first decade of the new century brings more dynamic web pages and applications. Users not only "browse" the Web but also contribute to it by producing and uploading their own content. The so-called Web 2.0 evolved and brought some major changes also in web development. The new way of interaction with Web calls also for the evolution of web languages with the main intention to formalize some of the already established good practices in web development.

In this paper we summarize some important novelties in new web standards and protocols. We concentrate primarily on the new version of HTML by presenting its new elements and extensions. We list also some most important JavaScript APIs which enable an entirely new way of web development by providing a browser-based database, information on geolocation, full-duplex communication between a browser and a server and other new exciting features. We point out also some new options in the CSS syntax. The latter introduces many new design options and also simplifies and standardizes some features that have been available before but suffered from poor browser support.

2 HTML5

HTML is in the continuous development since its introduction in the early 1990s. The majority of its features and functionalities have been defined through specifications, but some of them are also result of good development practices and the implementations of HTML in the popular browsers.

The actual HTML version - HTML4 - has been in use for almost a decade. According to W3C (*World Wide Web Consortium*) [1] one of the big disadvantages of HTML4 is that "*it does not provide enough information to build implementations that interoperate with each other and, more importantly, with a critical mass of deployed content. The same goes for XHTML1, which defines an XML serialization for HTML4, and DOM Level 2 HTML, which defines JavaScript APIs for both HTML and XHTML...*".

In order to provide better flexibility and interoperability of the HTML implementations and at the same time make web pages more interactive and offering better user experience, the development of HTML5 began within the WHATGW (*Web Hypertext Application Technology Working Group*) initiative and the W3C organization. The development is based on the study of the existing HTML4 implementations, good practices and analyses of the already deployed web content.

HTML5 [2] will be backward compatible with HTML4 and XHTML1, supporting both, HTML and XML *(eXtensible Markup Language*) syntax. It will also introduce new interfaces to support contemporary trends, such as rich internet applications (RIA). Currently, these interfaces depend strongly on the use of complex JavaScript code and proprietary plug-ins, such as Adobe Flash [3], Microsoft Silverlight [4] and Sun JavaFX [5]. The general idea suggested by the web developers is to implement the key functionalities for such interfaces in browsers themselves and remove the dependence on various proprietary plug-ins.

It is expected that HTML5 will achieve the candidate for recommendation status in 2012 and become a recommendation in 2022 [6]. Although the work on HTML5 will not be completed yet in the next few years, more and more of its functionalities are supported by web browsers [7].

2.1 Changes in the Language

The majority of web pages today uses common structures such as headers, footers and sidebars to denote the semantic structure of the page. Because HTML versions in use today do not provide special markup for this purpose, web developers use **div** and **span** elements, assigning them a unique id and/or arranging them into classes. HTML5 introduces a set of new elements, which allow semantic marking of the document structure. They represent more specific replacement for the general **div** and **span** elements. The current and the new approach to structuring a web page is shown in Figure 1.

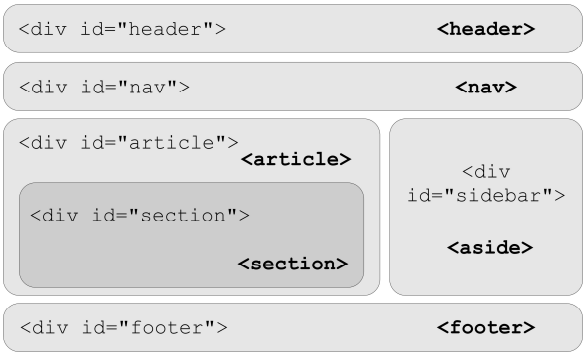

Figure 1. Current (using **div** elements) and the new approach (using new elements which are presented in bold text) to structuring a web page.

HTML5 introduces some other new elements, among which the most interesting are:

- elements that support multimedia and graphic content: **video**, **audio** and **canvas**. These elements are described in more detail in the following chapters;
- **embed** is used for the embedded content, handled by the plug-ins;
- elements for the display of quantities (**progress, meter, time** etc.);
- **ruby** for specifying annotations which are used in East Asian typographies.

HTML5 enforces strict separation of content and styling of a web page, which is manifested in the absence of the presentational attributes (e.g. **align**, **height**, **border**, **size**) and elements (e.g. **font**, **center**, **strike**, **u**). Page styling and design can only be done with CSS. Frames are also not supported in HTML5, due to their negative impact on the usability of a web page.

The elements **a** and **area** have a new attribute, called **ping**. It defines the URLs (*Uniform Resource Locators*), where a browser can send a notification when the user visits a hyperlink. The user tracking is currently mostly performed through the server-side redirects, which causes a long waiting for a selected page. The **ping** attribute enables the user agent to inform the user which addresses will be notified. In case of privacy concerns, a user can turn off the notifications while he or she can not influence the redirects.

The other novelties in HTML5 also include new global attributes, relations in elements **link** and **a,** events and many others.

2.2 Web Forms

Web forms enable interaction between a web client and a web server. The data in the forms entered by the user is sent to a server, which responds according to the received values (e.g. returns the result of a search). The choice of the widgets used in forms is, however, limited. One of the good web development practices is the validation of more complex data on the client side. The latter is preformed using JavaScript or any other client side scripting languages. To provide new form widgets and to avoid validation of data on both sides, several custom form widgets were developed that can be used through third-party JavaScript libraries.

The developer needs have encouraged the development of a new generation of Web forms, called Web Forms 2.0, which found their way into HTML5 specifications. New widgets are introduced as new values of the attribute **type** of the **input** element (**tel**, **search**, **url**, **email**, **datetime**, **date**, **month**, **week**, **time**, **datetime-local**, **number**, **range**, **color**). Besides new widgets, HTML5 also introduces the enhancements of the existing ones as well as the automatic validation of the entered data.

Regarding forms, two more novelties should be mentioned in this context. The first one has to do with the form elements, which do not have to be the descendants of the **form** *element* anymore. They can be placed anywhere in the HTML document instead and linked to a proper form by using their new **form** *attribute*. The second novelty concerns the use of the HTTP (*Hyper-Text Transfer Protocol*) protocol methods when sending data to the server. Beside GET and POST, in HTML5 also PUT and DELETE methods are supported.

```
<input type="date" />
```

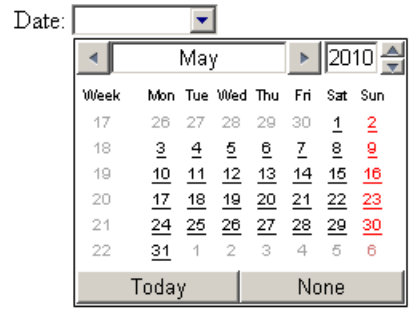

Figure 2. New **date** form widget.

2.3 Semantics

The HTML tags are intended for describing *how* to display the information and not *what* this information means. An important trend on the Web is the introduction of semantics into the resources. Besides defining the document structure in HTML5 the semantic markup is used also for recording microdata and assistive technologies for disabled users.

2.4 Microdata

The mechanism of microdata [8] enables the information in HTML documents, primarily intended for the end users (contact information, location information etc.), to be machine-readable and can therefore be used for automatic processing (e.g. for indexing, searching, storing, cross-referencing, analyzing etc.).

```
<div itemscope itemtype="http://example.org/band">
   <p>My name is
      <span itemprop='name'>Janko</span>.
   </p>
   <p>My band is called
      <span itemprop='band'>Four Parts</span>.
   </p>
   <p>I am
      <span itemprop='nationality'>Slovenian
      </span>.
   </p>
</div>
```

Figure 3. Example of using microdata.

The model of microdata is comprised of groups of properties named *items*. The items and their properties are presented in the context of the existing elements. An item can be created using the itemscope attribute in any element. Each *item* has its own type (itemtype), a global identifier (itemid) and a set of name-value pairs. The itemprop attribute can be used in any of the item's descendants to express a property, while the value of this attribute denotes the value of the property (Figure 3).

2.5 Accessibility

A significant portion of the Web content is currently inaccessible to users with disabilities, especially to those that depend on assistive tools, for example screen readers and Braille keyboards. The major problem for such tools is reading and interpreting the content which uses advanced, frequently updated user interfaces developed with the combination of the technologies AJAX (*Asynchronous JavaScript and XML*), HTML and JavaScript. The assistive technologies do not properly understand the roles, states and properties of such widgets and cannot follow the dynamically updating content on the web pages.

To overcome this problem, the Web Accessibility Initiative (WAI) started the ARIA (*Accessible Rich Internet Applications*) [9] project. ARIA specifications define a semantic model, which enables the authors to semantically describe the widgets and their behavior, document structure and the areas that will be updated. In this way, the assistive technologies would gather enough information to make advanced web applications usable also to people with disabilities.

```
<li role="menuitemcheckbox"
    aria-checked="true">Sort by Last Modified</li>
```

Figure 4. Example of using ARIA attributes. The list item element (**li**) is assigned a role (with the **role** attribute) of an element that acts as a checkbox item. Its initial property 'checked' is set to *true* (using **aria-*** set of attributes). Using JavaScript, the attribute could be changed according to user actions.

According to the semantic model, authors can appoint the *roles* the elements have on the page and define their *states* and *properties*. States and properties define the element's current semantic state and property and can therefore change from time to time while on the other hand a role remains unchanged when the page updates.

2.6 Audio and Video

Before HTML5, playing of audio and video contents in a browser has only been possible by the use of third-party plug-ins with the Adobe Flash being the leading technology. Besides introducing new elements for embedding audio and video contents in a web page, HTML5 also defines an interface for manipulation with such content without the need for plug-ins.

The general solution was the selection of a common format supported by all browsers. The basic requirement for such format was that it should not be proprietary. It should also have good compression and picture quality as well as small processing requirements. A hardware solution for the decoding should also exist. Instead of more popular MP3 and H.264, the W3C organization chose Ogg Vorbis for audio and Ogg Theora for video records, mainly because they are both license-free, which is not true for the rival MP3 and H.264.

Despite everything mentioned, the actual draft of the HTML5 specifications does not specify either a default format or any other formats that should be supported by the browsers. The final consensus between the W3C and browser vendors has never been reached. Therefore vendors individually decide on the formats built in their browsers.

Apple does not support the Ogg Theora in its Safari browser, due to the lack of the hardware acceleration and some uncertainties regarding the patents. On the other hand, Mozilla Foundation and Opera oppose to the use of H.264 due to the expensive licensing. Google's Chrome supports both Theora and H.264. The latter, however, cannot be included in the open source project Chromium and is supported mostly because it is used by Google's YouTube, which is the World's most important video portal.

2.7 Graphics

In the recent web development the graphic rendering on the web pages has only been possible with the aid of plug-ins, such as Flash or Silverlight. With the HTML5, the functionality needed for graphical rendering is implemented in browsers in the form of Canvas and SVG (*Scalable Vector Graphics*) technologies. The graphical elements are completely integrated into HTML and are also a part of the document object model (DOM). Their style can be defined through CSS and can be manipulated through the JavaScript.

Canvas enables dynamic rendering of graphics, e.g. graphs, bitmap images, animations and games, using scripting. The **canvas** element and its attributes **width** and **height** define a display region, which is then accessible through JavaScript code using Canvas API drawing functions.

Canvas does not differentiate between the objects in the graphic and does not contain the relations between these objects (such as DOM). The basic elements of canvas graphic are pixels. The rendered graphic is therefore final and cannot be rescaled. The individual graphic objects cannot be accessed, manipulated or interacted with. In order to make any changes, the whole graphic must be redrawn.

The basic Canvas API enables 2D rendering. 3D rendering will be possible using the web standard WebGL. Although the WebGL is still in the development, it is experimentally supported in most popular browsers.

HTML5 supports also browser-native rendering using SVG (*Scalable Vector Graphics*). SVG is an XML-based language for describing 2D vector graphics. Unlike Canvas, SVG enables rendering in high resolution at any level of magnification due to its vector nature. Today, SVG is used mostly for displaying static contents (maps, plans etc.) with the aid of browser plug-

ins. SVG is based on a special XML object model through which the individual graphic objects can be accessed and manipulated using JavaScript. SVG enables also interactivity by using event-handlers assigned to any SVG graphic object.

```
<canvas id="canvas" width="838" height="220"/>
<script>
  var canvasContext = document.
      getElementById("canvas").getContext("2d");
  canvasContext.fillRect(250, 25, 150, 130);
  canvasContext.beginPath();
  canvasContext.arc(360, 110, 100,
      Math.PI * 1/8, Math.PI * 12/8);
  canvasContext.lineWidth = 20;
  canvasContext.lineCap = 'round';
  canvasContext.strokeStyle = 'rgba(255,0,1,0.6)';
  canvasContext.stroke();
</script>
```

Figure 5. Canvas element and the manipulation of its content using JavaScript (adapted from [10]).

3 Presentation and Cascade Style Sheets (CSS)

The markup languages are used primarily to define basic structure of web documents and pages while on the other hand the final presentation and rendering are usually defined with CSS – a style sheet language which defines presentation semantics. The technique of designing web pages with CSS is almost 14 years old with the CSS 2.1 being the leading standard for the last 12 years. It is widely supported by the majority of web browsers and it evolved rapidly over the last decade. Along with the new markup languages the new CSS are being proposed as well. CSS3 is the new potential standard currently in the state of working draft or candidate recommendation. The massive specification of CSS3 standard [11] is divided into several modules which are developed individually with different progress speeds and dynamics. Various modules enable browser vendors to implement them incrementally. Several CSS3 modules are already supported by the majority of modern browsers. The new standard is completely backwards compatible with the addition of new properties and functionalities.

The new *Selectors* module introduces a variety of new methods for attaching elements to the corresponding style. The most exciting addition is the ability to select markup elements based on their placement in the DOM. The *:last-child*, *:nth-child(n)* and *:nth-last-child(n)* commands enable the targeting of elements based on their positions in a parent's list of child elements. For example the command *:nth-child(3n+2)* would mach a group of three elements after the second element. Other commands enable also the matching of elements which are checked, without children or elements that do not match the specified declaration. The markup elements can also be selected based on the existence of specific attribute and also just a part of an attribute. For example, the *img[alt*=”good”]* would select all images that have an *alt* attribute containing the word “good”.

```
<style type="text/css">
   img {
    border-width: 50px 50px 50px 50px;
    -webkit-border-image: url(logo.gif) 100% 100% 100%
     100% repeat repeat;
  }
</style>
<body>
  <div>
    <img src="logo.jpg">
  </div>
</body>
```

Figure 6. Use of one image as a border of the other image

A module on *Backgrounds and Borders* enables the use of multiple backgrounds which can be resized and positioned relatively or absolutely. It enables the re-use of images in several different contexts and more accurate filling of various areas. The module enables also the borders to use gradients, rounded corners, shadows and even border images. The *border-image* property allows an image file to be used as border of an object. An example is shown in Figure 6. The *gradient* property enables the borders or backgrounds to shift from one color to another programmatically.

The *Color* module defines different capabilities for setting all colors in the document. The new *rgba* command enables the specification of the color as well as the opacity of an element. Red, blue and green colors have to be defined with an integer value or with percentages while the alpha value should be between 0.0 and 1.0. For example, the alpha value of 0.8 defines the element with only 20% transparency.

Since now all web designers had to be aware of a set of the so called Web-safe fonts. The new *Fonts* module

allows the font file to be included as an external file and accessed through *font-family* property. The new *@font-face* rule allows the fonts to be called from an online directory, such as for example the command *@font-face { font-family: 'myFont'; src: url (../myFonts.ttf') format('truetype'); }*. If the command is not supported by the browser it reverts to the next specified font in the *font-family* property. The main issues with new font types are font licensing and the copyright. The embedded fonts can easily be downloaded from any page.

The two exiting new modules are also *Transitions* and *Animations* which enable the changes and movements of page elements in 2D or 3D space. With transitions the developer can specify the specific CSS property to animate from one state to another in a smooth transition. For example, a click on an image can trigger a change of the size of that image. The animations on the other hand enable the iteration of multiple CSS properties simultaneously for a number of times. The animation can be divided into stages using keyframes. Individual stages of elements between keyframes are approximated automatically by the browser.

4 APIs

HTML5 introduces a number of APIs that standardize features already present in today's browsers. Support for these features can be tracked on sites like [7] or [12].

4.1 Offline features

The Web has evolved from a Web of simple hyperlinked documents to a mashup of web applications. These applications are becoming more and more similar to desktop applications and therefore require similar facilities.

The first step in using local storage is to detect whether the user agent has a connection to the internet. W3C's Offline Web Applications [13] proposes usage of two events ("online" and "offline") dispatched to the Window object and an IDL (*Interface Definition Language*) attribute called "onLine" on the Navigator object. This allows an application to either actively decide whether to work with remote resources or be notified by the browser when state changes.

In recent past, using cookies was the only way to store data locally [14]. Although being some sort of a hack to the stateless nature of HTTP, cookies have been used widely to store small amounts of data on the client. Cookies have two downsides: limited storage (in the range of 4kB) and the fact that they are sent to the server for every request.

HTML5 introduces the idea of simple storage with the Web Storage draft [15]. Storage is accessed through an IDL attribute and supports simple key/value type storage. Two types of storage are considered in the draft: local and session storage. While local storage was conceived to store data persistently even after a user quits the browser, data in session storage expires right after the browser window closes. Furthermore, session storage also solves the problem of simultaneously running application instances because each window has a separate session storage.

This type of storage does not go without controversy. As it can contain sensitive information, it is very important for the user agents to manage data with care [16] – not allowing access from applications running under another domain and removing the data immediately upon application's request.

Web storage is implemented in most major browsers with a varying degree of adherence to standards [12].

There is much less compromise over more complex types of storage. SQLite [17] is an SQL compliant database that was popularized by Google as it was a part of Google Gears – an extension available to all major browsers that added support for some functionalities not found in user agents at that time. This type of transactional storage was also proposed to W3C for standardization, but the draft was frozen as it was too biased towards SQLite and no other implementations existed. Google Chrome and Opera support SQLite while Mozilla and Microsoft openly oppose this approach.

Another W3C draft for advanced storage is called Indexed Database API [18] (renamed from Web Simple DB [19] in late 2009). On an API level, this draft is perhaps similar to the Web Database draft, but it is actually an implementation of a key/value store. This specification goes much further than Web Storage as it also defines database, object stores (tables in DBMS – *Database Management System* lingo), indexes, relationships, cursors and transactions. In time of writing, no user agent implemented support for Indexed Database API although Google team started with implementation in March 2010 [20], and Mozilla is targeting their implementation for Firefox 4.0 [21].

4.2 Geolocation

HTML has become an increasingly popular technology on mobile devices and an answer to cross platform problems on mobile devices. As mobile operating system market is fragmenting (iOS, Android, Maemo/MeeGo, Symbian etc.), the HTML technology ecosystem is the only real counterweight to native applications.

Geolocation is one of the "killer technologies" in the mobile world. It sprung an ecosystem of applications and it also improves capabilities of existing applications. While geolocation has been available for different mobile platforms for quite a while, it is only now being standardized by W3C. When determining the location of a device, software does not rely on a single, but a range of technologies including GPS (*Global Positioning System*), Wi-Fi, RFID (*Radio Frequency Identification*) and mobile radio technologies.

```
//output last cached location
navigator.geolocation.getCurrentPosition(
    function(pos){
        alert("lat:" + pos.coords.latitude
            + ", long:" + pos.coords.longitude)
    }
);
```

Figure 7. Example of using geolocation API [22] through a *Navigator* object by retrieving last cached location.

Furthermore, the draft also includes the ability to automatically notify the application when a device location changes.

4.3 Web Workers

While JavaScript within browser supports asynchronous communication with servers, the user interface is always drawn and interacted with in a single thread. This means that while computationally intensive tasks can be done on the server without slowing down the user interface, there is no way to do them within a browser.

page1.html

```
var w = new Worker("worker.js");
w.postMessage("page1");
w.onmessage = function(evt){
    alert(evt.data);
}
```

worker.js

```
onmessage = function(evt){
    postMessage("Hello " + evt.data);
}
```

Figure 8. Sample code that shows communication between invoking page (top frame) and a worker (bottom frame) script. *postMessage()* method and *onmessage* event handlers are used for communication.

Web Workers draft is designed to cope with this issue [23]. The idea behind web workers is to allow background execution of long executing or computationally demanding tasks without interrupting or slowing down user's interaction with the browser.

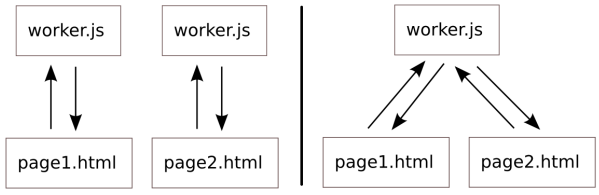

Figure 9. Both pages invoke "worker.js" as a *Worker* (left) which results in separate instances of scripts. The right image represents both pages communicating with a single instance of *SharedWorker*.

A thread (called a worker in web lingo) is created through a *Worker* object. The *Worker* object takes a single parameter: filename of the JavaScript to execute. This means that the code being executed in a separate thread is always contained in a separate file. While *Worker* objects always spawn new instances of worker objects, *SharedWorker* objects are shared between invoking scripts. This means that more than one script can connect to the same *SharedWorker* instance.

4.4 Two-way communication between a web browser and a server-side process

In the original model of the web, browser always requests a web page or parts of a web page. HTTP 1.1 specifications limit browsers to maximum of two simultaneous connections with a web server which enables faster loading and rendering of pages with multiple pictures or multimedia content. In the past, a "Comet" model has changed this classical model by keeping the one of the two connections alive permanently for real-time data exchange with the server. A web server can use this connection to push data to a browser without the browser to explicitly requesting it. In some cases just one long lasting connection can be used instead of two simultaneous connections. There have been several different implementations of this technique but all of them rely on AJAX and its *XMLHttpRequest* object.

The HTML5 *WebSocket* protocol [24] is a new uniform technique for pushing data to browser-based clients. It is a part of the HTML5 specification and it is intended to be used within scripts in web pages. It provides a full duplex connection without the need for multiple HTTP connections. A WebSocket request starts as a standard HTTP request from the client and then upgrades to the WebSocket protocol with a special initial handshake between the client and the server. The latter establishes a persistent HTTP connection which can be used for sending data in a full duplex mode. The communication is handled through JavaScript which requires the data to be in a text-based format.

WebSocket protocol places much less burden on web servers and enables the existing machines to handle twice the number of simultaneous connections. Another great benefit of the protocol is its ability to traverse firewalls and proxies. A WebSocket detects a proxy server and automatically establishes a tunnel to pass through the proxy by issuing an HTTP CONNECT statement. The latter requests the proxy to open a TCP/IP connection to a specific host. The similar solution works also for establishing secure connections (*Secure Sockets Layer*, SSL).

5 Conclusion

The general idea of HTML5 and other tools presented in this paper is the formal specification and the establishment of uniform solutions for technologies and functionalities which have already been in use through various hacks and plug-ins proposed by web developers. The majority of modern rich and interactive web designs was based on Adobe Flash technology which was supported by all major browser vendors. The Flash plug-in

offered an excellent support for multimedia content, especially animations and animated interfaces. HTML5 simplifies the implementation of such functionality through native browser support.

The new notable trend on the Web today is the introduction of semantics in web documents. The web content is shaped and designed primarily to be read and understood by people; therefore a computer cannot provide any extensive help by analyzing, searching and processing the data. The introduction of semantics will eventually lead to the third generation of Web, the so-called Semantic Web [25].

New development practices, rich web content and the need for semantics in web documents are already manifesting themselves in practice. Besides some changes in HTML syntax and vocabulary, the most important new features in HTML5 are the introduction of semantics in the form of microdata and ARIA attributes, support for RIA by bringing new form widgets, support for multimedia and dynamic graphic rendering.

References

[1] HTML5 differences from HTML4, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/html5-diff/, accessed 5. 7. 2010
[2] HTML5, A vocabulary and associated APIs for HTML and XHTML, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/html5/, accessed 5. 7. 2010
[3] Adobe Flash, http://www.adobe.com/flashplatform/, accessed 5. 7. 2010
[4] Microsof Silverlight, http://www.silverlight.net/, accessed 5. 7. 2010
[5] JavaFX, http://javafx.com/, accessed 5. 7. 2010
[6] WHATWG Wiki FAQ, http://wiki.whatwg.org/wiki/FAQ, accessed 5. 7. 2010
[7] WHATWG Wiki, Implementations in Web browsers, http://wiki.whatwg.org/wiki/Implementations_in_Web_browsers, accessed 5. 7. 2010
[8] HTML Microdata, W3C Working Draft, 24. 6. 2010, http://www.w3.org/TR/microdata/, accessed 5.7. 2010.
[9] Accessible Rich Internet Applications (WAI-ARIA) 1.0, W3C Working Draft, 15. 12 2009, http://www.w3.org/TR/wai-aria/, accessed 5. 7 2010
[10] HTML5, Web Development to the next level, html5rocks.com, http://slides.html5rocks.com/, accessed 12. 7. 2010
[11] Cascading Style Sheets, Current Work http://www.w3.org/Style/CSS/current-work, accessed 5. 7. 2010
[12] Offline Web Applications, W3C Working Group Note, 30. 5 2008, http://www.w3.org/TR/offline-webapps/, accessed 5. 7. 2010
[13] W. Peng, J. Cisna: HTTP cookies – a promising technology Online Information Review, Vol. 24, No. 2, 2000
[14] Web Storage, Editor's Draft, 15. 6. 2010, http://dev.w3.org/html5/webstorage/, accessed 5. 7. 2010
[15] A. Trivero: Abusing HTML 5 Structured Client-side Storage, http://trivero.secdiscover.com/html5whitepaper.pdf, accessed 5.7.2010
[16] Web Designer's Checklist, http://www.findmebyip.com/litmus/, accessed 5. 7. 2010
[17] SQLite Home Page, http://www.sqlite.org/, accessed 5. 7. 2010
[18] Indexed Database API, W3C Working Draft, 5. 1. 2010, http://www.w3.org/TR/IndexedDB/, accessed 5. 7. 2010
[19] WebSimpleDB API, W3C Working Draft, 29.9.2009, http://www.w3.org/TR/WebSimpleDB/, access. 5.7. 2010
[20] The Cromium Projects, Web Platform Status, Google inc., http://www.chromium.org/developers/web-platform-status#TOC-Indexed-Database-API-IndexedDB-form, accessed 5. 7. 2010
[21] A. Ranganathan: Beyond HTML5: Database APIs and the Road to IndexedDB, Mozilla.org, http://hacks.mozilla.org/2010/06/beyond-html5-database-apis-and-the-road-to-indexeddb/, accessed 5. 7. 2010
[22] Geolocation API Specification, W3C Working Draft, 7. 7. 2009, http://www.w3.org/TR/geolocation-API/, accessed 5. 7. 2010
[23] Web Workers, Editor's Draft, 25. 6. 2010 http://dev.w3.org/html5/workers/, accessed 5. 7. 2010
[24] The Web Sockets API, W3C Work. Draft, 22. 12. 2009, http://www.w3.org/TR/websockets/, accessed 5. 7. 2010
[25] T. Berners-Lee, J. Hendler, O. Lassila: The Semantic Web, Scientific American Magazine, 2001

Grega Jakus graduated from the Faculty of Electrical Engineering of the University of Ljubljana, Slovenia, in 2007. He is currently employed as a junior researcher in the Laboratory of Communication Devices at the same faculty. His research and development focus on machine translation algorithms, web technologies and telecommunication networks.

Matija Jekovec graduated from the Faculty of Computer and Information Science of the University of Ljubljana in 2004. He cofounded and is acting CTO of Web agency Klaro which deals with complete web solutions. He is responsible for application architecture in larger projects. In his spare time, he deals with client-side web technologies.

Sašo Tomažič is a Full Professor at the Faculty of Electrical Engineering of the University of Ljubljana. He is the Head of the Laboratory of Communication Devices and of the Chair of Telecommunications. His work includes research in the field of signal processing, security in telecommunications, electronic commerce and information systems.

Jaka Sodnik is an Assistant Professor at the Faculty of Electrical Engineering of the University of Ljubljana. He teaches subjects entitled Web Technologies and Terminals and Applications. His research interests include different aspects of acoustics, telecommunication networks, web technologies and human-computer interaction.